Relatório da Simulação de Sistema Operacional

# 1. Introdução

Este relatório apresenta a implementação de uma simulação de sistema operacional desenvolvida em linguagem C. São simulados conceitos fundamentais como gerenciamento de processos, escalonamento, alocação de memória, dispositivos de entrada/saída, sistema de arquivos e controle de concorrência.

# 2. Implementação dos Conceitos

• Gerenciamento de Processos: Utiliza uma estrutura para armazenar dados de controle dos processos (PCB), como estado, tempo de execução, prioridade e informações de memória.

• Escalonamento: Utiliza uma política FIFO (First In, First Out) para selecionar qual processo será executado.

• Execução Simulada: A execução dos processos é feita simulando 'quantums' de tempo.

• Gerência de Memória: Alocação de blocos contíguos de memória via estratégia First Fit.

• Dispositivos de E/S: Simulação de teclado e impressora com filas de requisições e interrupções.

• Sistema de Arquivos: Manipulação de arquivos em memória com permissões de leitura e escrita.

• Concorrência: Uso de semáforos para controlar o acesso a regiões críticas.

# 3. Limitações Encontradas

• A concorrência é sequencial e não usa threads reais.

• Não há preempção real nem escalonamento por prioridade.

• A memória não é compactada após desalocações.

• O sistema de arquivos não é persistente.

# 4. Demonstração de Execução

A simulação executa diversas funcionalidades de um sistema operacional:  
- Cria processos com diferentes características;  
- Aloca e libera memória;  
- Gerencia execução com escalonador FIFO;  
- Simula bloqueios e interrupções de E/S;  
- Manipula arquivos em memória;  
- Garante exclusão mútua com semáforos.  
  
A execução imprime mensagens de estado e mostra o fluxo de funcionamento do sistema.

# 5. Código Fonte Sem Comentários

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <stdbool.h>  
#include <time.h>   
  
  
  
  
typedef enum {  
 NOVO,  
 PRONTO,  
 EXECUTANDO,  
 BLOQUEADO,  
 FINALIZADO  
} EstadoProcesso;  
  
  
typedef struct {  
 int id;  
 EstadoProcesso estado;  
 char programa[50];   
 int tempo\_execucao\_restante;   
 int prioridade;   
 int tamanho\_memoria;   
 int endereco\_base\_memoria;   
} Processo;  
  
  
#define MAX\_PROCESSOS 100  
Processo \*fila\_prontos[MAX\_PROCESSOS];  
int num\_processos\_prontos = 0;  
  
  
Processo\* criar\_processo(int id, const char\* programa, int tempo\_execucao, int prioridade, int tamanho\_memoria) {  
 Processo\* novo\_processo = (Processo\*)malloc(sizeof(Processo));  
 if (novo\_processo == NULL) {  
 perror("Erro ao alocar memória para o processo");  
 exit(EXIT\_FAILURE);  
 }  
 novo\_processo->id = id;  
 novo\_processo->estado = NOVO;  
 strncpy(novo\_processo->programa, programa, sizeof(novo\_processo->programa) - 1);  
 novo\_processo->programa[sizeof(novo\_processo->programa) - 1] = '\0';  
 novo\_processo->tempo\_execucao\_restante = tempo\_execucao;  
 novo\_processo->prioridade = prioridade;  
 novo\_processo->tamanho\_memoria = tamanho\_memoria;  
 novo\_processo->endereco\_base\_memoria = -1;   
  
 printf("Processo %d ('%s') criado (Estado: NOVO)\n", novo\_processo->id, novo\_processo->programa);  
 return novo\_processo;  
}